German Amvrossiev

Automation of the Microsoft's Office programs

Espoo Metropolia University of Applied Sciences

Bachelor of Engineering

Information Technology

Thesis

22.5.2018

Metropolia

Tekijä Otsikko Sivumäärä Aika	German Amvrossiev Microsoft Office ohjelmien automatisointi 29 sivua + 1 liite 22.5.2018
Tutkinto	Insinööri (AMK)
Tutkinto-ohjelma	Tietotekniikka
Ammatillinen pääaine	Sulautettu tietotekniikka
Ohjaajat	yliopettaja Janne Salonen

Opinnäytetyön taustalla on kaksi itsenäistä projektia, jotka tehtiin kahdelle yritykselle. Ensimmäinen on Microsoft Outlookin lisäosa, jolla tallennetaan viestejä palvelimelle. Toinen on Excelille suunnattu makro, jolla helpotetaan tietojen siirtämistä taulukosta Word-pohjaiseen laskuun. Yritetään myös avata Microsoft Officen ja OpenOfficen historiaa.

Microsoft Office on tällä hetkellä maailman tunnetuin toimistojen työkalu, johon jokainen koneella työtä tehnyt on törmännyt. Se on myös nykyään täynnä ominaisuuksia, joilla saadaan työskentely mielekkäämmäksi.

Rasitusta aiheuttavat erityisesti copy-paste tyyliset työt, joissa ajatustyö korvautuu turhautuksella. Tässä työssä yritetään selvittää mahdollisuudet, joilla raskas työ voidaan automatisoida.

Avainsanat	Microsoft Office, Automatisointi, VSTO, OpenOffice, Makrot

Author Title Number of Pages Date	German Amvrossiev Automation of the Microsoft's Office programs 29 pages + 1 appendices 22 May 2018
Degree	Bachelor of Engineering
Degree Programme	Information Technology
Professional Major	Embedded technology
Instructors	Janne Salonen, Principal Lecturer

This thesis was inspired by two of my independent projects, created for the companies requiring automating their Microsoft Office programs. First one of which, is an Outlook plugin that saves mails to the server and the second, is an Excel macro that creates Word document invoices. Thesis will also try to scratch some history of the Microsoft Office and the OpenOffice.

Microsoft Office is the most popular software bundle for office use by far. Anyone, who worked on a computer from the early 2000's, is probably familiar with the product. Nowadays, office comes packed with features that make customer experience more pleasant.

Sometimes, user finds himself in a copy-pasting loop. This thesis will show, that there are ways to ease the user of such tasks and make the software do the heavy lifting for them.

Keywords	Microsoft Office, Automation, VSTO, OpenOffice, Macros

Contents

Appendices

Appendix 1. Excel macro

Abbreviations

OEM	Original equipment manufacturer
APP	Application
WYSIWYG	“What You See is What You Get”
RTF	Rich Text Format
VBA	Visual Basic for Applications
VB	Visual Basic
API	Application programming interface
AOO	Apache OpenOffice
IDE	Integrated development environment
VS	Visual Studio
VSTO	Visual Studio Tools for Office
UI	User Interface
Easter egg	Hidden secret
MAC	Macintosh

1 Introduction

Office work is monotonous. That's the first thing that might come to mind. Any work, if done again and again, will be. That is why considering automation is important. It covers old mechanical work with intellectual and leads to exponential growth in processing. The point is to eliminate repeat. How to automate the work of office solutions? It can be done by either using the tools provided by the software, e.g. macros or by creating custom software plugins using VSTO or other tools. Reading a manual and getting familiar with software might be a good idea.

In first chapters, we'll take a look at Microsoft Office and its open source competitors and their history. Later, we'll have a look at two of my personal projects that were done during the years 2016 and 2017. First is a larger functionality VSTO plugin for Office's Outlook 2016 done with Visual Studio and second is typical macro extension for Office's Excel and Word interoperability. We'll dig in to problems and challenges faced on the way as well as look at the positive sides that automation brought to the companies.

The major software used by almost all companies across the world is Microsoft's Office. It was developed by the Microsoft in 1988 and has been in heavy use since. There are alternatives to it most known of which is open source solution - OpenOffice. Microsoft's dominance on the office solution market is unanimously enormous. Microsoft Office is closed source but provides tools for modifying their products and offers macro automation. Microsoft Office has also the most well-known user interface. It's easy to get started and easy to learn. On the other hand, there is OpenOffice which tries to mimic Office's UI but comes free and has the same core functionality and products.

The reason why all the projects were made for the Microsoft's Office and not for the OpenOffice is simply, because none of the companies requesting automations were using it. My opinion on, which one is better is quite neutral, although I tend to lean more to Microsoft's proprietary solution, simply because It presented throughout my life and I got used to its UI. Many would agree, that they use Microsoft's Office for the same reason.

2 Microsoft Office

2.1 History

Microsoft Office was first announced in 1988 at ‘Computer Dealers' Exhibition’ – a computer exhibition in Las Vegas. It was a bundled software solution, containing Microsoft’s core components: Microsoft Word (made in 1983), Microsoft Excel (made in 1985) and Microsoft PowerPoint (made in 1987).

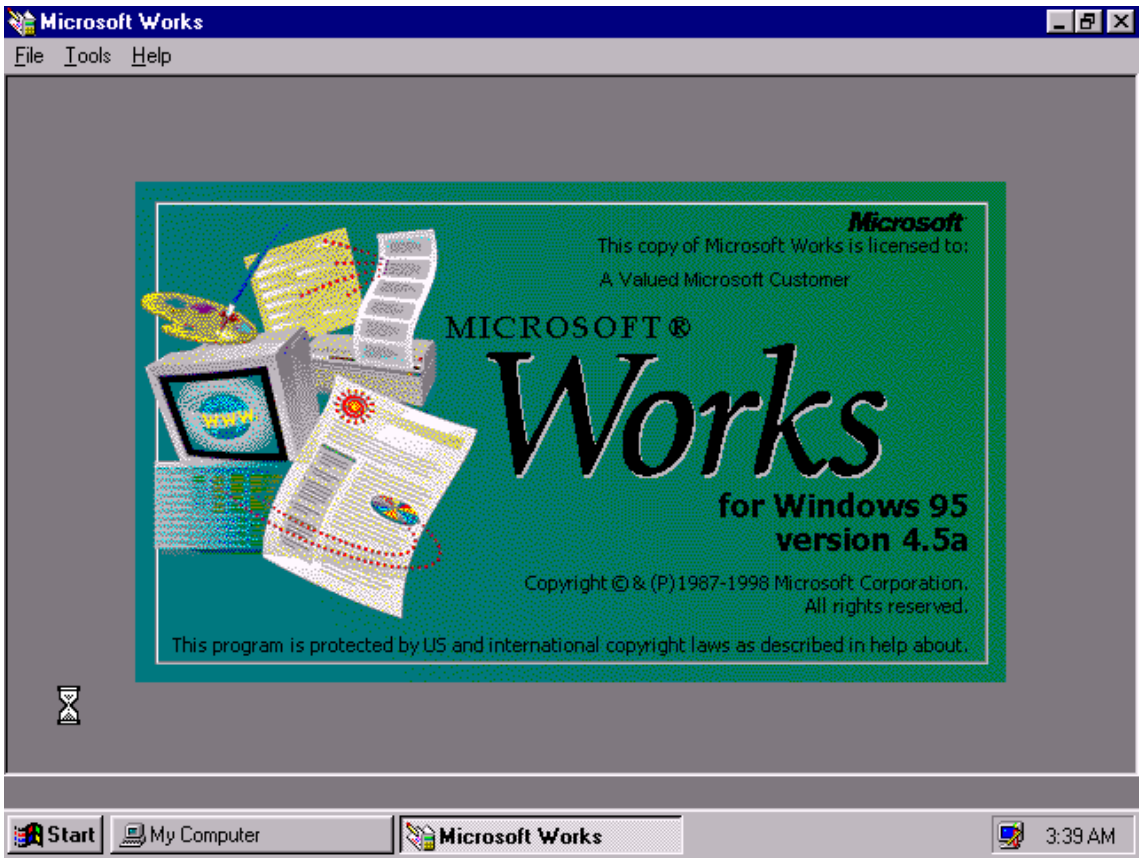

Figure 1. Microsoft Works for Windows 95.

Microsoft Office was introduced with and “lighten up” version of itself, called Microsoft Works. It had similar solutions but used less system memory and was originally designed for laptops. It gained popularity among low-cost office machines because of its low-cost price (about 40 USD retail, as low as 2 USD OEM). In late 2009 Works was merged into Office 2010 project as a “Starter” edition.

In 2011, Office 365 beta subscription was launched alongside Office 2010. Office 365 is a yearly subscription for Office products. It guarantees updates for a year, offers online services, like Office programs online availability, OneDrive and improved version of Skype – Skype for Business. It has also server and business services like Yammer and SharePoint. Subscription plan is the new selling strategy of Microsoft. Selling their Office solution in this way is considerably harder to pirate too and creates sustained income.

In July 2012 it was estimated, that over a billion of people use Microsoft Office.

There are many products today included with Microsoft Office and Microsoft 365, but I will be concentrating on original main three programs that are a necessity for almost any type of a customer. I will also look at other programs, that I have personal experience with. Some of a more complex Office's business programs will be out of this scope.

Office has a GitHub and Twitter accounts named "OfficeDev". It provides API for office developers. Judging by the Twitter account, the community was created somewhere in 2009. It is a good strategy towards creating the community of independent developers around their products. There are however, much better alternatives like OpenOffice, which are open source from the beginning and welcome developers worldwide.

2.1.1 Word

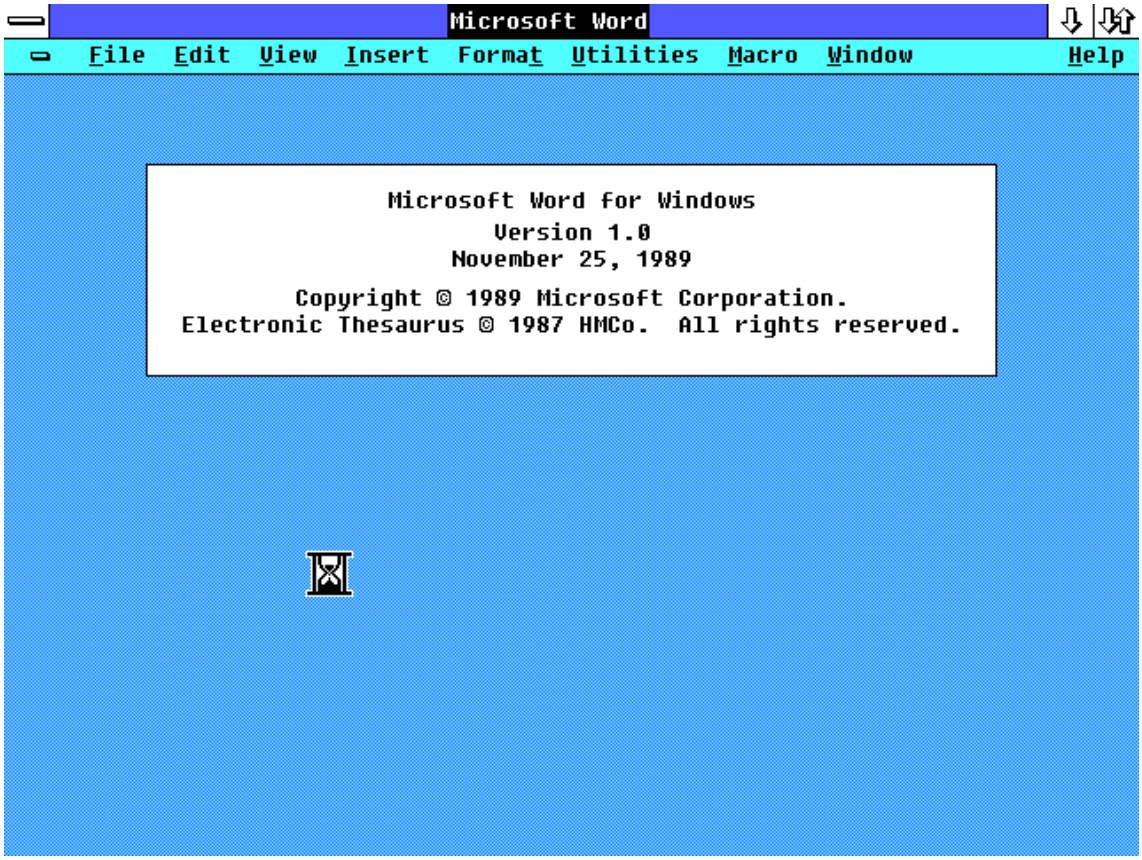

Figure 2. First version of Microsoft Word

Word is a text editing tool that comes with the Office software solution. It's mainly used for text editing in a "WYSIWYG" type environment. That meant, when applying changes to your document, you are expected to see changes immediately in a way the document is going to be published. It's used worldwide by offices, schools and personal purposes; for writing articles, spellchecking and printing.

Microsoft Word, originally named 'Multi-Tool Word', was one of the first applications on PC that used mouse. The mouse even came bundled with the software. Word originally came out in the year 1983. It was designed for Xenix and DOS systems, both discontinued operating systems. Subsequent versions had the support for OS Classic and Unix in 1985, ATARI ST in 1988, OS/2 and Windows in 1989, SCO Unix in 1994 and OS X in 2001.

Figure 3. Original Microsoft mouse from 1983.

Original Word was called “Multi-Tool Word” and was developed by Charles Simonyi and Richard Brodie, former Xerox employees. Newest features were mouse support - a then unpopular device, “undo” – button and an ability to view different font styling like bold, italic and underlined text. At the time, Word was competing with then popular word processing software – WordStar made by MicroPro International.

First “Classic Mac OS” – version was released in the year 1987 under version 3.0. It had numerous improvements and some new features, like RTF-support but was filled with bugs. The bugs were fixed in a free update, version 3.01. After MacWrite was discontinued in 1990’s, Word had no competition. It became dominant for the MAC’s operating systems.

In the year 1986, Word was ported on Atari ST, under the name Microsoft Write. No further updates were provided due to software piracy problem on Atari’s systems, that kind of led to the death of its computers.

With the release of Microsoft Windows, Word 3.0 was ported to on it. Because it was a port, it came with bugs and messy code. Microsoft was planning to rewrite Word for Windows and Mac OS from ground up but was abandoned because of the high resource demand for such task. Instead, resources were spent on some new features, like Auto-Correct and AutoFormat.

Mac OS version was highly criticized for many bugs, slow performance, clumsy controls and high memory use. Newer versions for Mac OS are ports of Windows versions with a mixture of Mac OS compatibility code.

Word's native file formats are ".doc", ".dot", ".docx" and ".docm". Binary formats (.doc) can be devided into compatibility modes (like 97-2004/2007) for older versions of Word. Doc – format is the legacy format, that is no longer encouraged to be used. It whey more and can't be translated anywhere outside Word.

The "x" at the end ".docx" indicates "Office Open XML international standard". It was established for compatibility possibilities across other word editors. It was implemented by Microsoft in Office 2007 and newer versions. Office Open XML uses XML – file type. That makes the file less prone code errors and makes it more compact. Files with "Office Open XML" - standard can be cross-opened with different editors. Office Open XML is not to be confused with OpenOffice.

The "m" at the end of ".docm" means the document has VBA macros embedded. User must pay extra caution when opening such kind of documents. VBA macros are popular playground for virus attacks.

The "t" at the end of ".doct" means it's a Word template. Word template contain information about document styles and formats.

Current version of Word is currently 16.0. It came with the new improvement – the Auto Save – feature. Auto Save – feature saves your work periodically. Similar feature was present in Google Docs. It saves your work on every new event.

2.1.2 Excel

Excel is spreadsheet editing software. Its main purpose is to calculate mass-data, create charts and analysis tables. Requires some learning to use program efficiently. Learning-curve is lenient for basic tasks thanks to user-friendly interface. To make full use of the program's potential, one must learn VBA, macros and Excel functions.

Excel comes with its collection of spreadsheet functions. User can create its own function by using functions provided or by writing own custom macros that extend the functionality of Excel with VBA.

Macros are written in VBA-language, that evolved from VB (Visual Basic), but can also be recorded with macro recorder. It's also possible to write add-ons that attach to Office programs at start. We will touch this subject in later chapters.

Excel was a successor to Microsoft's previous spreadsheet program Multiplan. Multiplan was not that popular on DOS, competing with then more popular software Lotus 1-2-3, but had commercial success on CP/M systems.

Excel hit the Macintosh market in 1985 and Windows market in November 1987. Lotus lost this battle, because was too slow to migrate to new Windows systems. Excel dominated the market and outsold the Lotus 1-2-3, releasing new version in every two years. Some noticeable improvements include VBA support in 1993, multi-object clipboard support in 2000, ribbon menu, Office Open XML and increased number of rows and columns in 2007 and x64 support in 2010.

Excel didn't come without its flaws, some of which were misleading statistics functions, mod function errors, date limitations and the Excel 2007 error – that had some float numbers resulting to a wrong round up.

Excel came quite often hidden with Easter eggs, most famous one of which is "The Hall of Tortured Souls". It could be triggered in Excel 95 by going to row 95 in a new spreadsheet, holding CTRL-ALT-SHIFT, and pressing the "Tech Support" button in the help menu. This triggered a mini Doom-kind-first-person game. The player could see Excel development team printed on the wall.

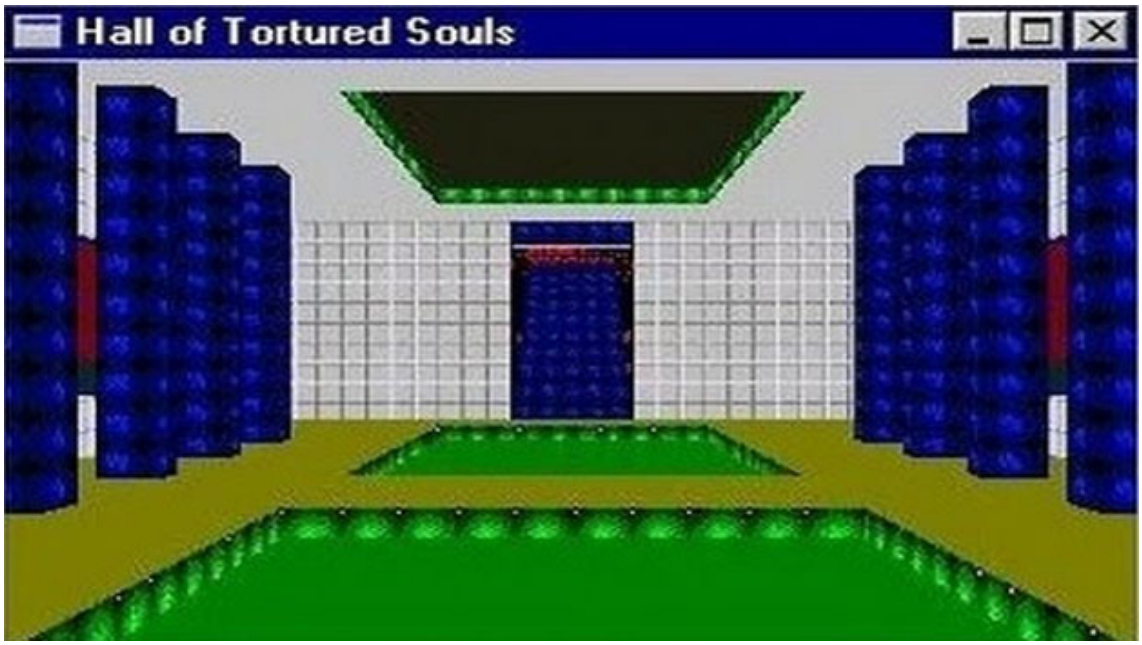

Figure 4. “Hall of Tortured Souls was the Easter egg found in Microsoft Excel 95. Excel had other Easter eggs hidden in the following versions.

2.1.3 PowerPoint

PowerPoint is a presentation program created by Robert Gaskins and Dennis Austin in 1987 by the company Forethought. Initially, PowerPoint was supposed to be released for Macintosh computers only but was acquired by Microsoft three months after the release. PowerPoint is currently holding 90% share of all the presentation software.

PowerPoint went under name “Presenter” during development but was changed during trademark registration to PowerPoint. It was supposed to be a software solution for “over-the-head” projector presentations. PowerPoint was planned for then unreleased first revision of Windows and Macintosh computers.

Microsoft wanted to develop the similar program. During preparational activities, Microsoft’s marketing division found out that a program of the same kind was already being developed by Forethought and that it was nearly completed. There was some skepticism, weather to acquire PowerPoint or not but Microsoft’s marketing department was able to convince Bill of the importance of overhead software. On July 30th, 1987 Microsoft had acquired Forethought Inc. for a total of $14 million dollars.

With the first release under the Microsoft's brand, no significant changes were made to Forethought's version. With second release color 35mm slides were added and with third release live video was added. Since the year 1993, PowerPoint was a part of Microsoft's Office bundle.

Nowadays, it's hard to neglect the impact that the PowerPoint had made to presentations world-wide. From business meetings to school seminars, PowerPoint is very powerful tool for empowerment of individual presenters and making visual impact on the audience, just like the name suggests.

2.1.4 Other interesting Microsoft Office programs

Other interesting software solutions that made into Microsoft's Office family are Access, Outlook and SharePoint. There are many software solutions that joined and left the Office family. Our scope is limiting to these three.

Outlook is widely used personal information manager. It includes mail, contacts, calendar and task managers. Originally it came out bundled with Exchange Server in 1993. It came for Windows, DOS and Macintosh.

Access is a database management system. It combines the relational Microsoft Jet Database Engine with a graphical user interface and software-development tools. Personal experience tells, that it is still used by companies to manage database data. It is good software to make connectivity with other Office products, like transforming an Excel spreadsheet data or for visual observing of a database structure.

SharePoint is a web-based collaborative platform. Usage varies between companies. Mostly it is used for web storage. SharePoint was developed during the Windows XP era. It went by the name "Office Server" and "Tahoe". It came from now discontinued Office WYSIWYG HTML editor – FrontPage. SharePoint is available in "Server" and "Online" versions. Online is Microsoft hosted, compared to Server. Server has more control and can be hosted on custom server. Applications include intranet, fileserver and file management. More features downloadable in Office's app store.

3 Open source solutions

3.1 History of the OpenOffice

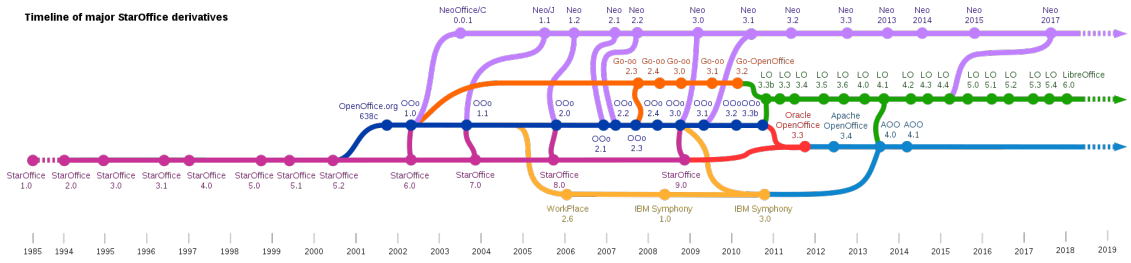

Figure 5. OpenOffice evolvement timeline

StarOffice originated in 1985 as StarWriter and StarDivision. They were bought by Sun Microsystems in 1999, that was later bought by Oracle Corporation in 2010. In 2000 source code of StarOffice was released, which spawned a new project called OpenOffice.org.

Oracle's OpenOffice project was closed in 2011 as part of the decision to turn OpenOffice.org into a "purely community-based project". There's a speculation, that it was the loss of mindshare. Developers working on the OpenOffice.org outside Oracle were concerned of Oracle's management and left to form TDF – The Document Foundation. It spawned yet another development branch called LibreOffice.

LibreOffice share many of the same features as OpenOffice but receives frequent updates. OpenOffice.org project was given out to Apache in 2011 and merged with IBM's Symphony in 2012. OpenOffice became Apache's top-level project. IBM ceased official participation by the release of AOO 4.1.1 in the end of 2014.

In September 2016, OpenOffice's project management committee chair Dennis Hamilton began a discussion of possibly discontinuing the project due to security issues.

OpenOffice is open source. That means, anyone who wants to contribute to the project, can register into developer's plan and develop new features. New features must be approved by the Apache's team. Source code can be downloaded from Apaches website.

Apache OpenOffice is a very large project with many components. It's mainly written in C++ but also uses Java, Perl, Python and other languages. Currently supported platforms include Linux variants, Windows, Mac OS X and OS/2. Source code folder sizes around 1,5 Gig.

4 Plugin for Outlook

4.1 Goal

The goal of this project was to create automated solution for saving incoming and outgoing mail if mail contained specific code in the subject area. The tool was requested by one of the Finnish jurisdictional companies. The solution was to create plugin for Microsoft Office's Outlook. Plugin operates on the client-side meaning, it will process mails downloaded from the server and won't be handling anything directly there.

4.2 Tools and programming language

Visual Studio was obvious IDE, because it features downloadable workload package for Office or SharePoint development. VS is also the IDE that I was most familiar with at the time. Office and SharePoint workloads contain necessary .NET frameworks, Office dev tools, analysis tools and tools for VSTO.

Most of the work was written in C#. Optionally, the project could have been written in VBA, but was settled on C# because of the familiarity with the language. Some project's files were auto-generated by IDE, e.g. form and project setting files. There is also an XML-file created for translating form dialogs to Finnish and English.

Originally, the add-on was intended for Microsoft's Office 2010. During the development, the company was switching to a newer version of Office – 2016, so the project was migrated to Office 2016 VSTO. No major changes were noticeable between 2010 and 2016 versions, except older version uses older .Net framework and newer is preferred to use newer (version 4 and higher). Not risking compatibility errors, the project was switched to use newer template of VSTO.

Git was used for version control. It is a free and open source distributed version control system.

I made 2 branches, of which first branch was the first prototype that was never used in production. It was the default “master” – branch. Second one was created about three months later. Company had requested more features into plugin, biggest of which was to save incoming mail. It’s named “test1” – branch. First and second branch describe the best of the two workloads of the separate time periods, master-branch being the prototype and test1-branch being the polished product.

The build-in build tool that came with Visual Studio was used. It’s a Visual Studio’s .NET C# Project that uses graphical forms to generate mandatory settings. The project can then be published from there. Publisher will generate installation package when publishing. In this project .Net Framework 4.6.1 was used.

During the second workload, the task was to create localizations and a user-friendly UI. A lot of time went into testing the project and many bugs were found during it. Some case-scenarios include, what happens if a user is disconnected from Internet? Things missing were different profiles for development and production, something similar that Apache’s Maven can do. No tests were written for the project. Comments were scattered throughout the code for the next developer. Comments and many of the methods were written in Finnish.

4.2.1 C# (C Sharp)

> “C# is a multi-paradigm programming language encompassing strong typing, imperative, declarative, functional, generic, object-oriented (class-based), and component-oriented programming disciplines. It was developed by Microsoft within its .NET initiative and later approved as a standard by Ecma (ECMA-334) and ISO (ISO/IEC 23270:2006). C# is one of the programming languages designed for the Common Language Infrastructure (CLI).” – Wikipedia

> “C# is a general-purpose, object-oriented programming language. Its development team is led by Anders Hejlsberg. The most recent version is C# 7.2, which was released in 2017 along with Visual Studio 2017.” – Wikipedia

4.2.2 .Net Framework

> ".NET Framework is a software framework developed by Microsoft that runs primarily on Microsoft Windows. It includes a large class library named Framework Class Library (FCL) and provides language interoperability (each language can use code written in other languages) across several programming languages. Programs written for .NET Framework execute in a software environment (in contrast to a hardware environment) named Common Language Runtime (CLR), an application virtual machine that provides services such as security, memory management, and exception handling. (As such, computer code written using .NET Framework is called "managed code".) FCL and CLR together constitute .NET Framework." – Wikipedia

4.3 Work flow

Figure 6. Version control of the Outlook add-in. Naming could have been better.

Workflow consists of two parts that are from different time periods. Major differences between "master" and "test1" branches are: in the master-branch work was mostly concentrated on saving the sent mail items compared to test1-branch where there is logic

added to save the incoming mail too. Second branch was created by the request from the company to expand the plugin. Other features mainly consist of additional logic to make the plugin more user-friendly and work seamlessly in different situations.

Second branch was a mixture of code from another plugin-project that saved incoming mail. At the time, it was created on par with “master” for testing purposes but had some major issues with incoming-mail-event being triggered multiple times or not at all.

4.3.1 Major difficulties

Biggest problems faced during development of Office’s plugin were the lack of support for developers. Searching the web gave little to no examples. “Getting started” – guides on this subject were close to none. Some questions existed on Microsoft’s forum pages, but had bland answers or overcomplicated ones. The best examples were found on the “Add-In-Express” site’s forum, where they gave out instructions on how to use their proprietary API. The explanations found there were helpful.

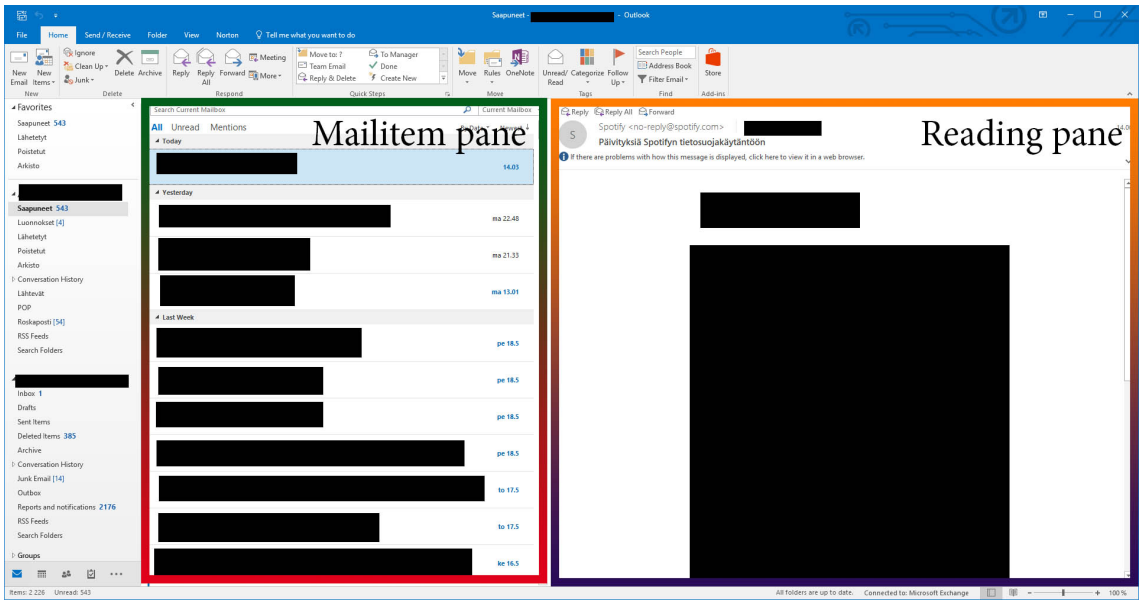

Figure 7. Cause of explorer-event-listener firing multiple times.

There was a problem with the explorer-event-listener. Explorer event fired twice on opening new mail-items. Because there was no way to see what triggered it, the only way to figure it out was guessing. The answer was never found online. Much later during the development it came to me, that the event was firing twice because of reading pane on the right. When user selects new mail from the mail-item pane, the event is first fired due to change there, and the second time due to change in the reading pane. Sometimes it event fired for the third time. Some extra logic was added to remove this problem, by comparing the mail-item to a previous one.

```
private void CurrentExplorer_Event() {
      _selectedFolder = Application.ActiveExplorer().CurrentFolder;
      _inBox = Application.Session.GetDefaultFolder(OlDefaultFolders.olFolder-
      Inbox);
      string _entryIDSelection = _selectedFolder.EntryID;
      //Check if the folder is incoming
      if (_entryIDSelection == _inBox.EntryID){
            try {
              //Inspect, if any windows are active?
              if (Application.ActiveExplorer().Selection.Count > 0) {
                  //Choose the first one
                  Object selObject = this.Application.ActiveExplorer().Selec-
                  tion[1];
                  //Check if it's mail?
                  if (selObject is Outlook.MailItem) {
                        _mail = (MailItem)selObject;
                        string ID = _mail.EntryID;
                        //Event repeats itself. Compare the e-mail to previous.
                        if (!ID.Equals(_IDPrevious) && !ID.Equals(_IDReceived)) {
                        saveMail();
                        _IDPrevious = ID;
                        }
                  }
              }
            }catch {/*Throws exceptions*/}
      }
}
```

Code example 1. Explorer-event-listener that has some extra logic to keep it from firing multiple times.

There was also a problem with finding out system's locale of a user's Office software. There was an attempt to get it by listening to the thread's "UI Culture". This was later resolved by just getting the system's (Windows) locale. Actual answer is still missing. Testing this was hard, because it required two versions of Office, one in Finnish and the other in English.

```
//office locale

//private static readonly string COUNTRY = Thread.CurrentThread.CurrentUICul-
ture.TwoLetterISOLanguageName;

//system locale

private static string COUNTRY = System.Globalization.CultureInfo.CurrentUICul-
ture.TwoLetterISOLanguageName;
```

Code example 2. Attempt on fetching the locale of the Microsoft Word. Upper one didn't work so it was settled to fetch the Windows locale.

Another problem was, that plugin had issues releasing resources. The issue was, when saving or deleting the mail-items, it couldn't use the mail-item because it was still locked by the plugin. Manual garbage collector releaser was implemented, shown in higher example. It had to run at the end of each event, releasing file locks.

```
private void emptyGarbage() {
            GC.Collect();
            GC.WaitForPendingFinalizers();
            GC.Collect();
}
```

Code example 3. Garbage collector for the Outlook add-in. Important for the long rning programs. Garbage collector controls the releasing of the resources.

4.4 Functionality of the plugin

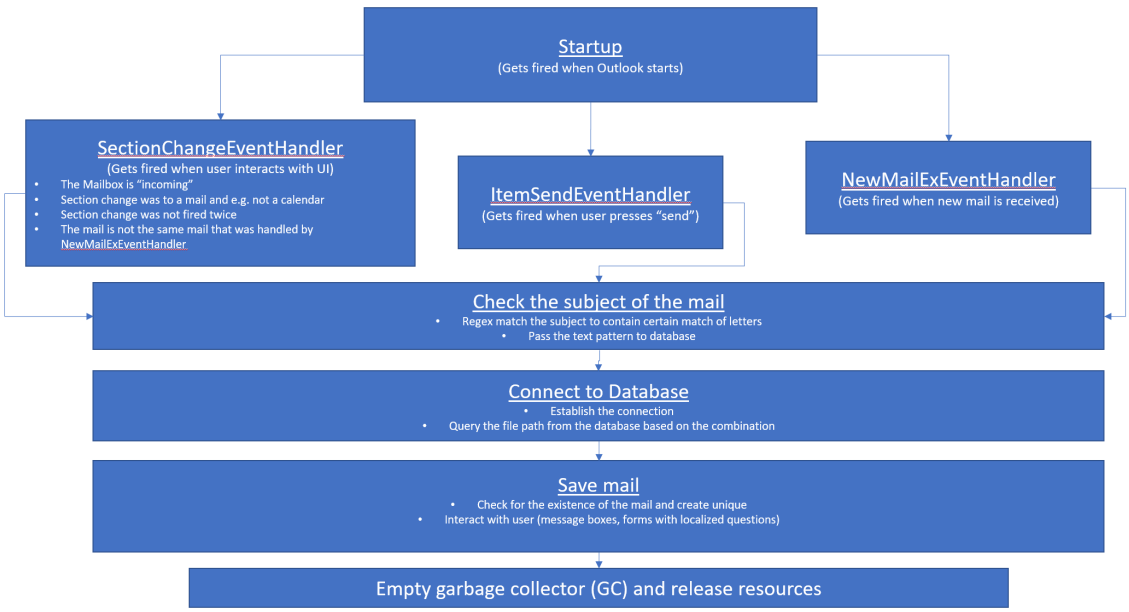

Figure 8. Abstraction of the Outlook plugin.

The plugin starts with Outlook. Outlook goes through a list of its COM-plugins at the start and fires them one by one. There is also deprecated event, that used to fire on a shut-down event of Outlook. The plugin template has the “Startup” and “Shutdown” (deprecated) events by default. Startup is used in the current plugin to launch tree interop event handlers: “SectionChangeEventHandler”, “ItemSendEventHandler” and “NewMailExEventHandler”.

```
private void ThisAddIn_Startup(object sender, System.EventArgs e) {
_currentExplorer = Application.ActiveExplorer();
_currentExplorer.SelectionChange += new Outlook.ExplorerEvents_10_Selection-
ChangeEventHandler(CurrentExplorer_Event);

Application.ItemSend += new Outlook.ApplicationEvents_11_ItemSendEv-
entHandler(SendButton);
Application.NewMailEx += new Outlook.ApplicationEvents_11_NewMailExEv-
entHandler(NewMailArrived);
}

private void ThisAddIn_Shutdown(object sender, System.EventArgs e) {
// Note: Outlook no longer raises this event. If you have code that
// must run when Outlook shuts down, see http://go.mi-
crosoft.com/fwlink/?LinkId=506785
}
```

Code example 4. Backbone of the project with the main three event listeners.

SectionChangeEventHandler gets triggered whenever there’s a section change in a UI. When user is selecting another mail-item, the event is going to be triggered and we can inspect what mail the user is currently viewing. There is a logic that will stop handling the event if the folder that user is viewing is not incoming-folder. Then, we figure out if the selected mail-item is new or unread and if it has the matching regex in the subject section. If so, the plugin will prompt user to save the mail to the directory, that is queried from the company’s database.

There’s going to be the following logic fired after the user says - “yes”:

- The plugin will try to find the folder path that was queried from the database and create it if it doesn’t exist.

- Then, it will generate a filename for the mail, that has all the non-compatible filename characters removed and check, whether the mail with the same name already exists.

- If so, it will compare mail-item's content and save the it if they have different content. New unique name will be created. If user prompts "no" everything is ignored.

- There is an extra logic to close all the opened mail-items in that directory opened by a current user. The plugin will prompt for that. Closing the opened mail-items is required to unlock them. Asynchronous editing of a file is forbidden in Windows.

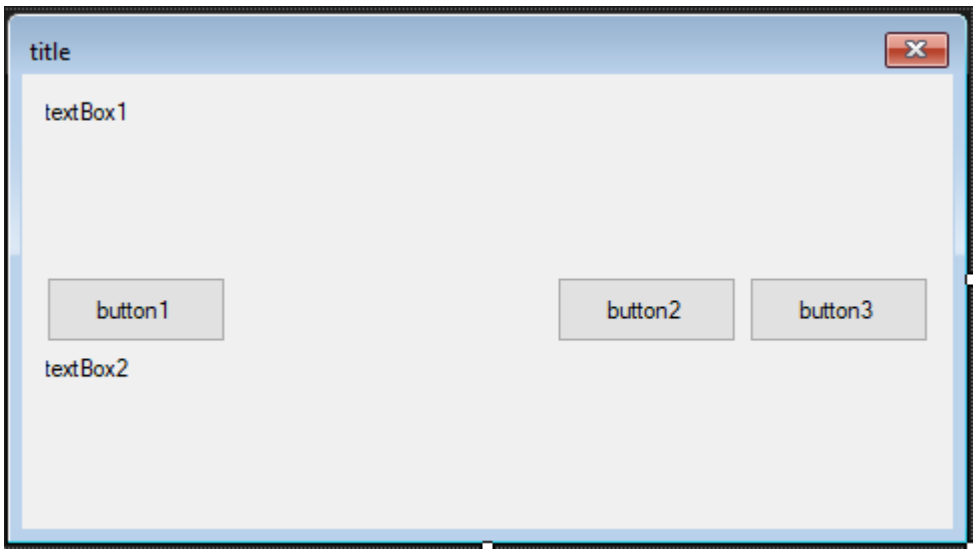

Figure 9. Custom Form class that was created for querying user to close all opened mailitems. All the texts are placeholders. Actual data comes from localized XML elements. "textbox2" is hidden and will become visible if "button1" is pressed, similar to "describe more" button.

ItemSendEventHandler and NewMailExEventHandler both work the same way as SectionChangeEventHandler. The only difference is, the way they are triggered. ItemSendEventHandler fires when the user sends mail and NewMailExEventHandler fires when the new mail is received. Because the software was developed on the client's side, new mail will not be processed, if the user has his mail client off, i.e. the NewMailExEventHandler will not be triggered if the Outlook is off. That created the need for the SectionChangeEventHandler implementation, so that important mail is not missed.

5 Excel and Word interoperability

5.1 Goal

There was a Finnish weight loss company that had a request for a bill creation automation. They used Excel spreadsheet for holding the customer data and were billing them by creating invoices with Word. This was done manually by hand. They wanted to have that automated, so that when a customer row on a spreadsheet was selected there could be a button to create an invoice based on row's data.

5.2 Tools and programming language

5.2.1 VBA

> "VBA is an implementation of Microsoft's event-driven programming language Visual Basic 6, which was discontinued in 2008, and its associated integrated development environment (IDE)." – Wikipedia.

> "Visual Basic for Applications enables building user-defined functions (UDFs), automating processes and accessing Windows API and other low-level functionality through dynamic-link libraries (DLLs). It supersedes and expands on the abilities of earlier application-specific macro programming languages such as Word's WordBASIC. It can be used to control many aspects of the host application, including manipulating user interface features, such as menus and toolbars, and working with custom user forms or dialog boxes." – Wikipedia.

5.2.2 Macros

Macros are specific input sequence that will generate the expected output result. The reason for their existence is to release the user from repeating redundant bulks input, i.e. combinations of keystrokes.

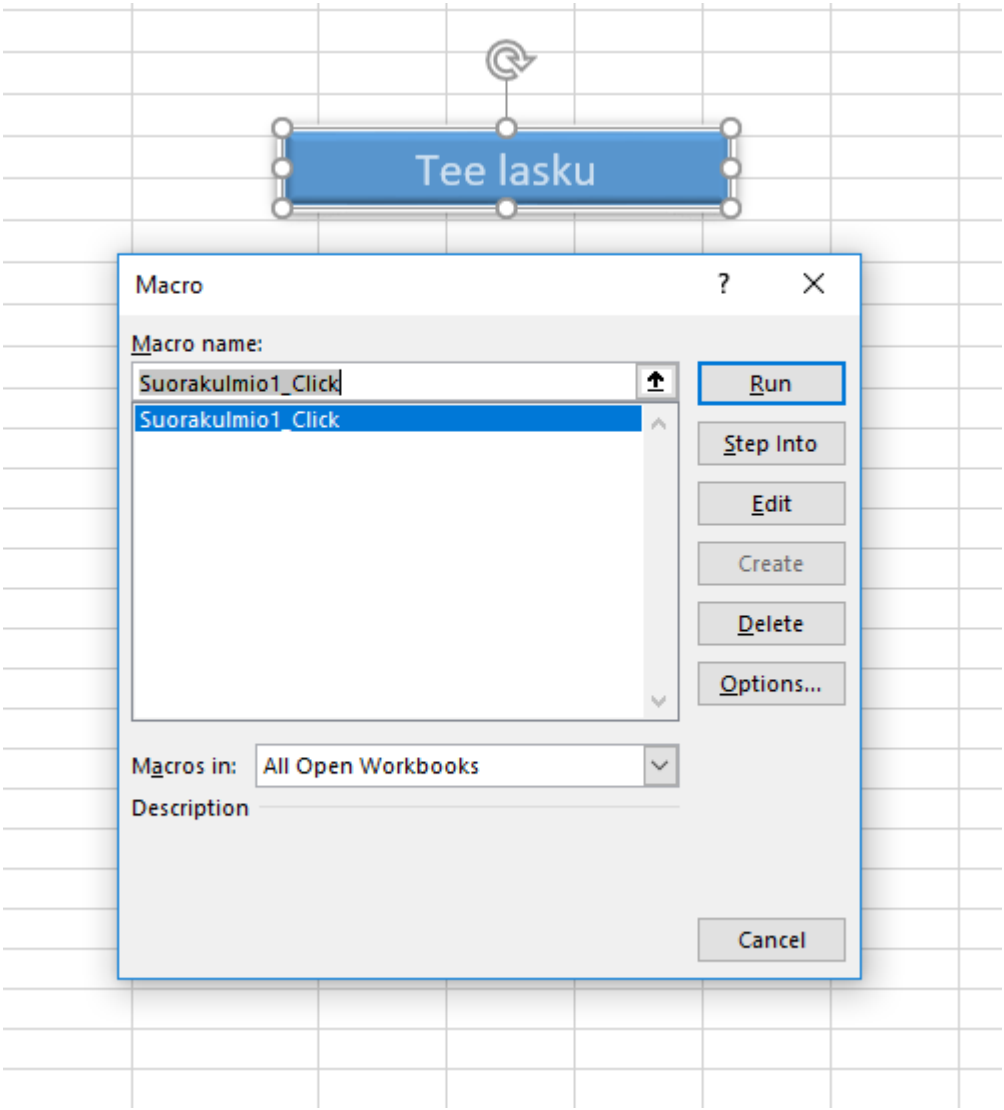

Figure 10. Macro editor in Excel.

Macros are personal and can be considered a programmable button that, when pressed, run a customized personal task. There is a user-friendly interface for creating macros in Office programs. No coding is required, because macros can be recorded with a macro-recorder. It means, the user presses the record-button, then inputs the sequence and releases record. The software will then generate the macro based on the history of the events that happened during the recording. Macros can also be assembled, by stacking pseudo-code function-blocks into a sequence, just like in Access the actual SQL-queries can be done.

5.3 Excel macro

The company had two documents: a template of the invoice on a Word document and an Excel spreadsheet of all the customer contacts. On a template, there is some fields that need to change for each customer, e.g. name, the amount of money they owe and the personal reference number. Also, there's the invoice number, that needs to increment each time the bill is generated. Company also requested, that the word file is not further processed after the Excel generates it, so that the accountant can double-check it before sending it to the customer.

Macro, made with VBA, is quite simple. On the UI there's two features added: "selection highlight" and "make-the-bill-button". Well call it – the make-button. Word document had to be changed to ".docm" (macros enabled word document) and create some placeholders for the incoming data from Excel. Macro works as follows:

- Selecting some row of data, i.e. customer, will highlight it unless the column is empty. That is when all the data columns are stored into variables.

	Nimi	Puh.	e-mail	Summa	Viitenumero			Laskunumero
2								
3	Mikko Mallikas			227,00 €	123333			9004580025493
4	Anna Mikkeli			299,00 €	123133			
5	Zak Hunter			50,00 €	123322			
6	Masen Hasen			20,00 €	111111			
7	Han Solo			35,00 €	322332			
8	Zara Doger			5,00 €	133221			
9								
10								

Figure 11. Selected row is highlighted and name, sum, reference number and invoice numbers are stored to variables.

- Upon pressing the make-button, Excel tries to open the Word document, find all the placeholders and replace them with variables. Some variables are generated on the backend, e.g. date and taxational sum.

- Lastly, the invoice-number-column's value is incremented by one in the Excel spreadsheet.

- None of the documents are saved and are left for the user to decide whether to close or save them.

Saajan tilinumero Mottagarens kontonummer	**IBAN** Nordea Pankki [redacted]	**BIC** [redacted]
Saaja Mottagare	[redacted]	Laskun numero 9004580025494
Maksajan nimi ja osoite Betalarens namn och address **Zak Hunter** **Allekirjoitus Underskrift**		**Viitenumero Ref.nr** 123322
Tililtä Från konto nr		**Eräpäivä** 25.5.2018 **Euro** 50

Figure 12. The automatically created invoice.

There is a major issue with this because user can accidentally save the Word document, which will override the template. The concern was passed on to the user. He, on the other hand, made sure to make copies of the original templates, in case of accidentally overwriting the templates.

Another concern is, that this project felt like it wasn't nearly finished when being pushed to production. Error-handling is close no none and the macro isn't user-friendly. This whole thing is also highly dependent on Microsoft keeping updates to legacy VBA and I'm not completely sure, if this will run on any other platform than Windows. Since, the project is small, it doesn't matter. Fixing this thing wouldn't bring that much of a trouble.

5.4 Result

Feedback from the company was extremely positive. It saved a lot of time and they were extremely happy with the product. The tool is still in use to this day. No new feature requests were asked for. Good example, of a simple thing making a huge impact.

6 Last thoughts

To this day, Microsoft Office is popular among office tools across the world. This has something to do with people's habits and familiarity with legacy Microsoft's products and a fairly long lifespan (around 30 years) of Office software on the market without competition. Many grew up using Microsoft's products.

Fact is, that Microsoft is nowadays charging outrages prices for their software. Currently, they charge anywhere from a 100 USD to 500 USD for the latest version, and that is not counting the new subscription plan the Office 365 has to offer. Subscriptions range from 60 to 80 USD per year, and the licenses we are talking about, are for one pc or user only. This must do something with Microsoft forcing their products on their own platform – Windows, making it easy to raise the price and keep competition away. Unless customer is a Linux user, there's a good chance he will choose Microsoft Office on his Windows operating system, because of the familiarity with the product. He will ignore the price, because there's nothing to compare it to.

Open source office tools have opened a good alternative to proprietary office tools. They tend to offer similar features, required for a normal user, and are free. Normal consumer has an option, but often chooses the more expensive one for no reason. If the need is for some basic WYSIWYG text editing and maybe a spreadsheet calculation, one doesn't need to stick with the Microsoft Office.

Even though this might sound like a lot of hate towards Microsoft Office, it's actually not. Microsoft has a great product that had us typing reports at school, to spreadsheets at work with great comfort. There is a feeling that the product has come a long way since it's early existence and made some good features along the way. The reason why everyone is still using the product, must mean that it is not bad.

References

1. How It's Made: Tower Defense, Excel 2010 game. YouTube video. https://www.youtube.com/watch?v=SFzi0Xs5_vw.

2. Microsoft Excel. Wikipedia article. https://en.wikipedia.org/wiki/Microsoft_Excel

3. Microsoft Word. Wikipedia article. https://en.wikipedia.org/wiki/Microsoft_Word

4. Microsoft PowerPoint. Wikipedia article. https://en.wikipedia.org/wiki/Microsoft_Powerpoint

5. Microsoft Office. Wikipedia article. https://en.wikipedia.org/wiki/Microsoft_Office

6. Microsoft Office. Wikipedia article. https://en.wikipedia.org/wiki/Microsoft_Works

7. Office 365. Wikipedia article. https://en.wikipedia.org/wiki/Office_365

8. Dev Hunter - a hidden game in Excel 2000. YouTube video. https://www.youtube.com/watch?v=PGZfuwsvIFQ

9. Excel functions (by category). Documentation. https://support.office.com/en-us/article/excel-functions-by-category-5f91f4e9-7b42-46d2-9bd1-63f26a86c0eb

10. Math errors found in Excel 2007. Web article. https://www.accountingweb.com/technology/excel/math-errors-found-in-excel-2007

11. Office Developer - repository. GitHub repository. https://github.com/OfficeDev

12. Office Developer – twitter account. Twitter account. https://twitter.com/OfficeDev

13. Top 5 Free Microsoft Office Alternatives. YouTube video. https://www.youtube.com/watch?v=58TK_3TyEfQ

14. Apache's OpenOffice home page. Home page. https://www.openoffice.org/

15. Apache OpenOffice. Wikipedia article.
https://en.wikipedia.org/wiki/Apache_OpenOffice

16. LibreOffice vs OpenOffice App Suite Comparison / Review 2017. YouTube video.
https://www.youtube.com/watch?v=uH9x8gWFCPo

17. OpenOffice [Software Review]. YouTube video.
https://www.youtube.com/watch?v=PS5F3cuI0hk

18. StarOffice. Wikipedia article.
https://en.wikipedia.org/wiki/StarOffice

19. C# Guide. C# manual, by Microsoft.
https://docs.microsoft.com/en-us/dotnet/csharp

20. Programming paradigm. Wikipedia article.
https://en.wikipedia.org/wiki/Programming_paradigm#Support_for_multiple_paradigms

21. Add-in-express homepage blog. Web blog.
https://www.add-in-express.com/creating-addins-blog/2015/01/23/working-outlook-attachments-programmatically/

22. Visual Basic for Applications. Wikipedia article.
https://en.wikipedia.org/wiki/Visual_Basic_for_Applications

23. Microsoft Office store. Online shop.
https://products.office.com/en/buy/office

24. What Is a .DOCX File, and How Is It Different from a .DOC File in Microsoft Word? Web review.
https://www.howtogeek.com/304622/what-is-a-.docx-file-and-how-is-it-different-from-a-.doc-file-in-microsoft-word/

25. Microsoft Word 1.x (Windows). Microsoft software archive.
https://winworldpc.com/product/microsoft-word/1x

26. Microsoft Office review. WordPress blog.
https://biotfanime.wordpress.com/2012/11/19/microsoft-office/

27. Microsoft mouse. SourceForge Blog.
https://sourceforge.net/blog/today-tech-may-2/

28. Easter eggs in Microsoft products. Wikipedia article.
https://en.wikipedia.org/wiki/List_of_Easter_eggs_in_Microsoft_products

29. The Hall of Tortured Souls. Web article.
https://www.geek.com/hall-of-tortured-souls/

30. Git homepage. Homepage.
https://git-scm.com/

Excel Macro

```
Public nimi As String
Public telefon As String
Public majl As String
Public summa As Currency
Public viite As Variant
Public laskunum As Variant

Sub Suorakulmio1_Click()

    Dim objWord, objDoc, objSelection, currentDirectory
    targetList = Array("")
    currentDirectory = ActiveWorkbook.Path

    Set objWord = CreateObject("Word.Application")

    Set objDoc = objWord.Documents.Open(ActiveWorkbook.Path & "\wordTestExample.docm")

    Set wdFind = objWord.Selection.Find

    objWord.Visible = True

    objWord.Activate

    Set myRange = objWord.ActiveDocument.Content

    myRange.Find.Execute FindText:="[NIMI]", ReplaceWith:=nimi, Replace:=wdReplaceAll
    myRange.Find.Execute FindText:="[YHT]", ReplaceWith:=Round(summa, 2), Re-
place:=wdReplaceAll
    myRange.Find.Execute FindText:="[HINTA]", ReplaceWith:=Round(summa * 100 / 124, 2),
Replace:=wdReplaceAll
    myRange.Find.Execute FindText:="[VEROT]", ReplaceWith:=summa - Round(summa * 100 /
124, 2), Replace:=wdReplaceAll
    myRange.Find.Execute FindText:="[LASTCALL]", ReplaceWith:=DateAdd("d", 3, Date), Re-
place:=wdReplaceAll
    myRange.Find.Execute FindText:="[RAND]", ReplaceWith:=laskunum, Re-
place:=wdReplaceAll
    myRange.Find.Execute FindText:="[VIITE]", ReplaceWith:=viite, Replace:=wdReplaceAll

    objWord.ActiveDocument.Sections(1).Headers(wdHeaderFooterPrimary).Range.Find.Execute
FindText:="[DATE]", ReplaceWith:=Date
    objWord.ActiveDocument.Sections(1).Headers(wdHeaderFooterPrimary).Range.Find.Execute
FindText:="[LASTCALL]", ReplaceWith:=DateAdd("d", 3, Date)
    objWord.ActiveDocument.Sections(1).Headers(wdHeaderFooterPrimary).Range.Find.Execute
FindText:="[RAND]", ReplaceWith:=laskunum

    laskunum = laskunum + 1
    Range("H" & 3).Value = laskunum
    'MsgBox (laskunum)

End Sub
```

```
Private Sub Worksheet_SelectionChange(ByVal Target As Range)

Dim rownumber As Integer
rownumber = ActiveCell.Row

    If Application.Intersect(ActiveCell, [Headers]) Is Nothing Then
        If ActiveCell.Value <> "" Then
            Range("a1:h5000").Interior.ColorIndex = xlNone
            Range("a" & rownumber & ":e" & rownumber).Interior.Color = RGB(255, 255, 9)
            Range("H" & 3).Interior.Color = RGB(255, 255, 9)
            nimi = Range("A" & rownumber)
            summa = Range("D" & rownumber)
            viite = Range("E" & rownumber)
            laskunum = Range("H" & 3)
            'MsgBox (laskunum)
        End If
    End
```